Cell-Probe Lower Bounds for Succinct Partial Sums

Mihai Pătraşcu
AT&T Labs

Emanuele Viola*
Northeastern University

October 21, 2009

Abstract

The partial sums problem in succinct data structures asks to preprocess an array $A[1\,..\,n]$ of bits into a data structure using as close to n bits as possible, and answer queries of the form $\textsc{Rank}(k) = \sum_{i=1}^{k} A[i]$. The problem has been intensely studied, and features as a subroutine in a number of succinct data structures.

We show that, if we answer $\textsc{Rank}(k)$ queries by probing t cells of w bits, then the space of the data structure must be at least $n + n/w^{O(t)}$ bits. This redundancy/probe trade-off is essentially optimal: Patrascu [FOCS'08] showed how to achieve $n+n/(w/t)^{\Omega(t)}$ bits. We also extend our lower bound to the closely related $\textsc{Select}$ queries, and to the case of sparse arrays.

1 Introduction

Consider an array $A[1\,..\,n]$ of bits. Can we preprocess this array into a data structure of size $n + r$ bits, for small redundancy r, which supports rank queries $\textsc{Rank}(k) = \sum_{i=1}^{k} A[i]$ efficiently? One can also consider the query $\textsc{Select}(x)$, which asks for k such that $\textsc{Rank}(k-1) < x \leq \textsc{Rank}(k)$.

The problem of supporting rank and select is the bread-and-butter of succinct data structures. It finds use in most other data structures (for representing trees, graphs, suffix trees / suffix arrays etc), and its redundancy / query trade-off has come under quite a bit of attention.

Rank already had a central position in the seminal papers on succinct data structures. Jacobson [Jac89], in FOCS'89, and Clark and Munro [CM96], in SODA'96, gave the first data structures using space $n + o(n)$ and constant query time. These results were slightly improved in [Mun96, MRR01, RRR02].

In several applications, the set of ones is not dense in the array. Thus, the problem was generalized to storing an array $A[1\,..\,n]$, containing m ones and $n - m$ zeros. It is generally assumed that $n < m \cdot \lg^{O(1)} m$; outside this range, the problem essentially goes outside the

*Supported by NSF grant CCF-0845003.

world of succinct data structures, as the lower bounds for predecessor search apply [PT06]. The optimal space is $B = \lg \binom{n}{m}$. Pagh [Pag01] achieved space $B + O(m \cdot \frac{(\lg\lg n)^2}{\lg n})$ for this sparse problem. Golynski et al. [GGG+07] essentially gained a log factor for $m \approx n$, achieving space $B + O(n \cdot \frac{\lg\lg n}{\lg^2 n})$. Subsequently, Golynski et al. [GRR08] achieved space $B + O(m \cdot \frac{\lg\lg n \cdot \lg(n/m)}{\lg^2 n})$, extending the log-factor improvement even to sparse arrays.

A qualitative improvement to these bounds is obtained in [Păt08], which shows an exponential dependence between the query time and the redundancy. Specifically, with query time $O(t)$, the achievable redundancy is $r \leq n/(\frac{\lg n}{t})^t$. This improved the redundancy for many succinct data structures where rank/select queries were the bottleneck.

Given the surprising nature of this improvement, a natural question is whether we can do much better. In this paper, we rule out this possibility:

Theorem 1. *In the cell-probe model with words of $w \geq \lg n$ bits, a data structure that supports* Rank *or* Select *queries over an array of n bits requires at least $n + n/w^{O(t)}$ bits of space, where t is the cell-probe complexity.*

The lower bound matches the upper bound [Păt08], except for the difference between $(\lg n)^t$ and $(\frac{\lg n}{t})^t$. This difference is inconsequential up to $t = \lg^{1-\varepsilon} n$. However, if we want a polynomially small redundancy (say, less than n^α, for constant $\alpha < 1$), the upper bound says that $t = O(\lg n)$ is sufficient, whereas the lower bound says that $t = \Omega(\lg n / \lg\lg n)$ is necessary. Closing this gap appears to be a difficult problem.

For sparse arrays, we obtain the following result:

Theorem 2. *In the cell-probe model with words of $w \geq \lg n$ bits, a data structure that supports* Rank *or* Select *queries over an array of n bits containing exactly m ones requires at least $\lg \binom{n}{m} + m/w^{O(t)}$ bits of space, where t is the cell-probe complexity.*

This matches the upper bound when $n \leq m \cdot \lg^{O(1)} m$ and $t \leq \lg^{1-\varepsilon} n$. As noted above, for $n \gg m \cdot \operatorname{poly}\log(m)$, a better lower bound can be obtained from the (colored) predecessor problem [PT06, PT07].

Our lower bounds hold in the cell-probe model [Mil99], which is a nonuniform generalization of the Word RAM. The Word RAM model with words of $w = \Omega(\lg n)$ formalizes the unit-cost operations of modern programming languages (random access to memory, arithmetic operations, etc), and is broadly assumed for the analysis of algorithms.

1.1 Related work

Much work in lower bounds for succinct data structures has been in the restricted model known as systematic model. In this model, the array A must be represented as is, but in addition the data structure may store an index of sublinear size, which the query algorithm can examine at no cost. See [GM03, Mil05, GRR08, Gol07] for increasingly tight lower bounds in this model. Note, however, that in the systematic model, the best achievable redundancy with query time t is $\frac{n}{t \cdot \operatorname{poly}\log n}$, i.e. there is a linear trade-off between redundancy

and query time. This is significantly improved by the (non-systematic) upper bounds in [Păt08]. Therefore, previous lower bounds qualitatively miss the nature of this improvement.

In the unrestricted cell-probe model, the first lower bounds were shown by Gál and Miltersen [GM03] in 2003. These lower bounds were strong, showing a linear dependence between time and redundancy: $r \cdot t = \Omega(n/\lg n)$. However, the problem being analyzed is somewhat unnatural: the bound applies to polynomial evaluation, for which nontrivial succinct upper bounds appear unlikely. Their technique, which is based on the strong error correction implicit in their problem, remains powerless for "easier" problems.

A significant break-through occured in SODA'09, when Golynski [Gol09] showed a tight lower bound of $r \cdot t^2 = \Omega(n)$ for the problem of storing a permutation π and querying both $\pi(\cdot)$ and its inverse $\pi^{-1}(\cdot)$. Due to the particular attention it pays to inverses, it is unclear how Golynski's technique could generalize to problems like rank.

We also note that [Vio09a] proves lower bounds for succinct data structures in the bit-probe model ($w = 1$). The argument in [Vio09a] breaks down when reading $w = \Omega(\log n)$ bits, although some of the techniques in that paper are useful in the cell-probe setting, cf. [Vio09b].

In this paper, we make further progress on getting lower bounds for natural problems, and analyze some of the central problems in succinct data structures. It is reasonable to hope that our lower bound technique will generalize to many other problems, given the many applications of rank and select queries.

2 Proof of The Main Result

In this section, we prove a space/time trade-off for the rank problem in a vector of n random bits.

2.1 The Top-Level Induction

We will now lower bound the complexity of RANK in an array A, where every $A[i]$ is an independent, uniform bit.

Imagine that the queries $\text{RANK}(0), \ldots, \text{RANK}(n-1)$ are divided into k blocks of $\frac{n}{k}$ consecutive queries (the remainder is ignored if k doesn't divide n). Let $Q_\Delta = \{\Delta, \frac{n}{k}+\Delta, 2\frac{n}{k}+\Delta, \ldots\}$ be the set containing the Δ-th query (counting from zero) in each block. For a set Q of queries, let $\text{Ans}(Q)$ be the vector of answers to the queries in Q. We treat $\text{Ans}(Q)$ as a random variable, depending on the random choice of the input.

To support the induction in our proof, we augment the cell-probe model with *published bits*. These bits represent a memory of bounded size (P bits) which the query algorithm can examine *at no cost*. Like the regular memory (which must be examined through cell probes), the published bits are initialized at construction time, as a function of the input. Observe that if we have n published bits, the problem can be solved with zero query complexity.

Our proof will try to publish a small number of cells from the regular memory which are accessed frequently. Thus, the complexity of many queries will decrease by at least one.

The argument is then applied iteratively: the cell-probe complexity decreases, as more and more bits are published. If the cell-probe complexity hits zero, while less than n bits are published, we have a contradiction.

Let $\text{Probes}(q)$ be the set of cells probed by query q; this is a random variable, since the query can be adaptive. Also let $\text{Probes}(Q) = \bigcup_{q \in Q} \text{Probes}(q)$.

Our main technical result is captured in the following lemma, the proof of which appears in the next section:

Lemma 3. *Let A be uniform in $\{0,1\}^n$. Assume that a data structure can answer* RANK *queries over A using n memory bits and $P = o(n)$ published bits. Break the queries into $k = \gamma \cdot P$ blocks, for a large enough constant γ. Then:*

$$\Pr_{A,q \in [n]} \Big[\text{Probes}(q) \cap \text{Probes}(Q_0) \neq \emptyset \Big] \;=\; \Omega(1).$$

The lemma suggests that $\text{Probes}(Q_0)$ is a good set of cells to publish, since a constant fraction of the queries probe at least one cell from this set.

Completing the proof is now easy. If the data structure has redundancy r, begin by publishing some arbitrary $P_0 = r$ bits; this ensures that there are at most n bits in regular memory, as required in the lemma.

In step $i = 0, 1, 2 \ldots$, we let $k_i = \gamma \cdot P_i$, and publish the cells in $\text{Probes}(Q_0)$. Each cell requires w bits for its contents and $O(\lg n)$ bits for its address, bringing the number of published bits to $P_{i+1} \leq P_i + k_i \cdot t \cdot O(w) = O(P_i w^2)$. Here, t stands for the cell-probe complexity of a query, and $t \leq \lg n$, since otherwise there is nothing to prove.

After publishing these bits, the cell-probe complexity of $\text{RANK}(k)$ decreases by $\Omega(1)$, on average over uniform A and k. Since the average complexity cannot go below zero, the maximum number of iterations that we are able to make must be $O(t)$. The only reason we may fail to make another iteration is a violation to the lemma's condition $P = o(n)$. Thus, $P_{O(t)} = r \cdot w^{O(t)} = \Omega(n)$. We obtain the desired trade-off: $r \geq n / w^{O(t)}$.

2.2 Correlation of Threshold Queries

In this section, we make the first step towards proving Lemma 3. Let P and k be as in the lemma's statement, and assume for contradiction that $\Pr_{A,q \in [n]}[\text{Probes}(q) \cap \text{Probes}(Q_0) \neq \emptyset] \leq \varepsilon^4$, for a small enough constant ε. We thus know that a random query is very unlikely to probe cells in $\text{Probes}(Q_0)$.

By averaging, there exists a $\Delta \in \{(1-\varepsilon^3)\frac{n}{k}, \ldots, \frac{n}{k} - 1\}$ such that $\Pr_{A,q \in Q_\Delta}[\text{Probes}(q) \cap \text{Probes}(Q_0) \neq \emptyset] \leq \varepsilon$. We are only going to concentrate on the queries Q_0 and Q_Δ.

Intuitively speaking, our contradiction is found as follows. The answers to queries Q_0 must be encoded in the cells $\text{Probes}(Q_0)$, and the answers to Q_Δ in $\text{Probes}(Q_\Delta)$. By assumption, $\text{Probes}(Q_\Delta)$ is almost disjoint from $\text{Probes}(Q_0)$. But the answers $\text{Ans}(Q_0)$ and $\text{Ans}(Q_\Delta)$ are highly correlated, since they ask for partial sums that are only $\varepsilon^3 \cdot \frac{n}{k}$ away (remember that we chose $\Delta \geq (1-\varepsilon^3)\frac{n}{k}$). Thus, if the two sets of answers are written in

disjoint sets of cells, a lot of entropy is being wasted, which is impossible for a succinct data structure.

Unfortunately, the random variable $\mathrm{Ans}(Q_i)$ turns out to be rather unwieldy, since it has superconstant entropy (we cannot afford the loss of an ε fraction of the queries in Q_Δ). Instead, we will define a boolean version of our query. Let $X(i)$ be the variable $\textsc{Rank}(i + \frac{n}{k}) - \textsc{Rank}(i)$; this is the sum of a block of $\frac{n}{k}$ elements following i.

Now define $T(i)$ as the indicator of whether $X(i)$ exceeds its mean: $T(i) = 1$ if $X(i) \geq \frac{n}{2k}$, and $T(i) = 0$ otherwise. Extend the definition of T to a set of queries, $T(\{q_1, q_2, \dots\}) = \big(T(q_1), T(q_2), \dots\big)$. Note that we can compute $T(i)$ from the answer of $\textsc{Rank}(i)$ and $\textsc{Rank}(i + \frac{n}{k})$. Then, $T(Q_\Delta)$ is a function of $\mathrm{Ans}(Q_\Delta)$. Thus, we may work with $T(Q_0)$ and $T(Q_\Delta)$ instead of $\mathrm{Ans}(Q_0)$ and $\mathrm{Ans}(Q_\Delta)$. To implement our intuition, the first thing to prove is that $T(Q_0)$ and $T(Q_\Delta)$ are well correlated:

Lemma 4. $\mathbf{H}\big(T(Q_0)\big) = k - o(k)$, *but* $\mathbf{H}\big(T(Q_0) \mid T(Q_\Delta)\big) = O(\varepsilon k)$.

Proof. We deduce this lemma from the following:

Fact 5. *If Z_N is the sum of N independent bits, $\frac{2}{\sqrt{N}} Z_N - \frac{N}{2}$ converges uniformly to the standard normal distribution $\mathcal{N}(0,1)$: the pointwise (ℓ_∞) distance between the two cumulative distribution functions is $O(\frac{1}{\sqrt{N}})$.*

This is an immediate instantiation of the Berry-Esseen theorem.

Let $Q_0 = \{q_0, q_1, \dots\}$ and $Q_\Delta = \{q'_1, q'_2, \dots\}$ (sic). Note that the values $T(q_i)$ are independent, since the $X(q_i)$'s are sums of different blocks of elements. Thus, $\mathbf{H}\big(T(Q_0)\big) = k \cdot \mathbf{H}\big(T(q_i)\big)$. We claim that $\Pr[X(q_i) \geq \frac{n}{2k}] = 1 - o(1)$. This follows from Fact 5, since the normal has probability $\frac{1}{2}$ above the mean. We conclude that $\mathbf{H}\big(T(q_i)\big) = 1 - o(1)$ so $\mathbf{H}\big(T(Q_0)\big) = k - o(k)$.

For the second part of the lemma, we expand $\mathbf{H}\big(T(Q_0) \mid T(Q_\Delta)\big)$ using the chain rule for entropy:

$$\mathbf{H}\big(T(Q_0) \mid T(Q_\Delta)\big) \;=\; \sum_i \mathbf{H}\big(T(q_i) \mid T(q_0), \dots, T(q_{i-1});\; T(q'_1), T(q'_2), \dots\big)$$

Conditioning on less variables may only increase the entropy; thus, we may bound:

$$\mathbf{H}\big(T(Q_0) \mid T(Q_\Delta)\big) \;\leq\; \mathbf{H}\big(T(q_0)\big) + \sum_{i \geq 1} \mathbf{H}\big(T(q_i) \mid T(q'_i)\big)$$

To analyze $\mathbf{H}(T(q_i) \mid T(q'_i))$, we observe that $X(q_i)$ and $X(q'_i)$ are sums of Δ common elements, plus $\frac{n}{k} - \Delta$ unique elements for each. Formally, we can write $X(q_i) = A + B$ and $X(q'_i) = A + C$, where A is the sum of $\Delta \geq (1 - \varepsilon^3)\frac{n}{k}$ elements, B and C are the sum of $\frac{n}{k} - \Delta \leq \varepsilon^3 \cdot \frac{n}{k}$ elements, and A, B, C are independent.

From Fact 5 applied to B and C, we conclude that $\Pr\big[|B - \mathbf{E}[B]| \geq \varepsilon\sqrt{\frac{n}{k}}\big] = O(\varepsilon)$. Applying the fact to A, we have $\Pr\big[|A - \mathbf{E}[A]| > \varepsilon\sqrt{\frac{n}{k}}\big] = 1 - O(\varepsilon)$.

By union bound, with probability $1 - O(\varepsilon)$, A deviates from its expectation by more than $\varepsilon\sqrt{\frac{n}{k}}$, but B and C deviate by at most $\varepsilon\sqrt{\frac{n}{k}}$. Thus, with probability $1 - O(\varepsilon)$, $T(q_i) = T(q'_i)$. We conclude that $\mathbf{H}\big(T(q_i) \mid T(q'_i) = O(\varepsilon)$. □

2.3 An Encoding Argument

In this section, we complete the proof of Lemma 3. We show that, if $\Pr_{A,q\in Q_\Delta}[\text{Probes}(q) \cap \text{Probes}(Q_0) \neq \emptyset] \leq \varepsilon$, we can encode the input A via a prefix-free code using strictly less than n bits on average, an information-theoretic contradiction.

Our encoding needs a careful definition of the intuitive notion of "the contents of the cells Probes(Q)." Define the footprint Foot(Q) of a query set Q by the following algorithm. We assume the published bits are known in the course of the definition. Enumerate queries $q \in Q$ in increasing order. For each query, simulate its execution one cell probe at a time. If a cell has already been included in the footprint, ignore it. Otherwise, append the contents (but not the address) of the new cell in the footprint.

The crucial property of this definition is that Foot(Q) is a string of exactly $|\text{Probes}(Q)| \cdot w$ bits.

Now observe that Ans(Q) is a function of Foot(Q) and the published bits. Indeed, we can simulate the queries in order. At each step, we know how the query algorithm acts based on the published bits and the previously read cells. Thus, we know the address of the next cell to be read. We can check whether this cell appeared previously the footprint (since inductively we know the addresses of all previous cells). If not, we read the next w bits of the footprint, which are precisely the contents of this cell, and continue the simulation.

With the definition of the footprint, we can now describe our encoding for the array A:

1. the published bits (P bits). Denote these bits by the random variable $\mathcal{P}$.
2. the identity of the (few) queries in Q_Δ that touch Probes(Q_0), i.e. the set $Q^\star = \{q \in Q_\Delta \mid \text{Probes}(q) \cap \text{Probes}(Q_0) \neq \emptyset\}$. We can represent this using $O\big(\lg \binom{|Q_\Delta|}{|Q^\star|}\big)$ bits. By submodularity, the average length of this component is on the order of:

$$\mathbf{E}\Big[\lg \binom{k}{|Q^\star|}\Big] \;\leq\; \lg \binom{k}{\mathbf{E}[|Q^\star|]} \;\leq\; \lg \binom{k}{\varepsilon k} \;=\; k \cdot O(\varepsilon \lg \tfrac{1}{\varepsilon})$$

3. the footprint Foot($Q_\Delta \setminus Q^\star$). This uses $\mathbf{E}\big[|\text{Probes}(Q_\Delta \setminus Q^\star)|\big] \cdot w$ bits. Observe that from the published bits, the identity of $Q^\star$, and this footprint, one can decode the answers Ans($Q_\Delta \setminus Q^\star$).
4. the entries of $T(Q_\Delta)$ that are not already known. Note that whenever two consecutive queries from Q_Δ are outside $Q^\star$, the entry of $T(Q_\Delta)$ that talks about their difference is known, since Ans($Q_\Delta \setminus Q^\star$) are known. Thus, at least $k - 2|Q^\star|$ entries in $T(Q_\Delta)$ are known; this component writes down the rest, in order. The expected size is $O(\varepsilon k)$ bits.
5. the value of $T(Q_0)$, encoded optimally given $T(Q_\Delta)$. Using any efficient code (like Hamming coding), this takes $\mathbf{H}\big(T(Q_0) \mid T(Q_\Delta)\big) + O(1)$ bits on average. By Lemma 4, this is $O(\varepsilon k)$ bits.
6. the footprint Foot(Q_0), encoded optimally given knowledge of $T(Q_0)$ and the published bits. This takes $\mathbf{H}\big(\text{Foot}(Q_0) \mid T(Q_0), \mathcal{P}\big) + O(1)$ bits, using an efficient code. We can rewrite $\mathbf{H}\big(\text{Foot}(Q_0) \mid T(Q_0), \mathcal{P}\big) \;=\; \mathbf{H}\big(\text{Foot}(Q_0), T(Q_0) \mid \mathcal{P}\big) - \mathbf{H}\big(T(Q_0) \mid \mathcal{P}\big)$.

But remember that $T(Q_0)$ is a function of $\mathrm{Ans}(Q_0)$, which can in turn be recovered from $\mathrm{Foot}(Q_0)$ and $\mathcal{P}$. Thus, the first term is $\mathbf{H}\big(\mathrm{Foot}(Q_0), T(Q_0) \mid \mathcal{P}\big) = \mathbf{H}\big(\mathrm{Foot}(Q_0) \mid \mathcal{P}\big) \le \mathbf{H}\big(\mathrm{Foot}(Q_0)\big) \le w \cdot \mathbf{E}\big[|\mathrm{Probes}(Q_0)|\big]$.

For the second term, we write $\mathbf{H}\big(T(Q_0) \mid \mathcal{P}\big) = \mathbf{H}\big(T(Q_0), \mathcal{P}\big) - \mathbf{H}\big(\mathcal{P}\big) \ge \mathbf{H}\big(T(Q_0)\big) - P \ge k - o(k) - P = \Omega(k)$. Therefore, the size of this component is $w \cdot \mathbf{E}\big[|\mathrm{Probes}(Q_0)|\big] - \Omega(k)$.

7. all cells outside $\mathrm{Probes}(Q_0) \cup \mathrm{Probes}(Q_\Delta \setminus Q^\star)$, included verbatim with w bits per cell. From the footprints included above, the addresses of the cells $\mathrm{Probes}(Q_0)$ and $\mathrm{Probes}(Q_\Delta \setminus Q^\star)$ are known. Thus, we know exactly which cells are included in this component, and we do not need to spell out their addresses. This component takes $n - w \cdot \mathbf{E}\big[|\mathrm{Probes}(Q_0)| + |\mathrm{Probes}(Q_\Delta \setminus Q^\star)|\big]$ bits on average.

Observe that this encoding includes the published bits and *all* cells in the memory (though the cells in $\mathrm{Probes}(Q_0)$ and $\mathrm{Probes}(Q_\Delta \setminus Q^\star)$ are included in a compressed format). Thus, all n queries can be simulated, which allows us to reconstruct the entire array A. Thus, we have a correct encoding of A.

Let us now calculate the average size of the encoding, summing up all components. Observe that items 3 and 6 contribute the footprint sizes: $w \cdot \mathbf{E}\big[|\mathrm{Probes}(Q_\Delta \setminus Q^\star)|\big]$, respectively $w \cdot \mathbf{E}\big[|\mathrm{Probes}(Q_0)|\big]$. But these terms are cancelled because the cells in the footprint are not included in item 7, which takes $n - w \cdot \mathbf{E}\big[|\mathrm{Probes}(Q_0)| + |\mathrm{Probes}(Q_\Delta \setminus Q^\star)|\big]$. Items 1, 2, 4, and 5 total $k \cdot O(\varepsilon \lg \frac{1}{\varepsilon})$ bits.

Thus, the encoding size is $n + k \cdot O(\varepsilon \lg \frac{1}{\varepsilon}) - \Omega(k)$, where the $\Omega(k)$ saving is from item 6 (where we took out the entropy of $H(T(Q_0))$ from the footprint). For a small enough constant ε, the negative term is twice the positive one, and the encoding has size $n - \Omega(k)$. We have reached a contradiction.

3 Extensions

We now extend our lower bound to both Rank and Select queries, in arrays that contain a prescribed number, m, of ones. The optimal space in this case is $\lg \binom{n}{m}$. Observe that setting $m = \frac{n}{2}$, we also get a lower bound for Select in an unrestricted bit vector, since $\lg \binom{n}{n/2} = n - O(\lg n)$.

Our strategy is to reinterpret the Rank and Select problems in a common partial-sums framework. Let $B[0\,..\,N-1]$ have *integer* elements, chosen independently from some distribution $\mathcal{D}$; the optimal average space is $n \cdot \mathbf{H}(\mathcal{D})$. Consider the partial sums query $\textsc{Sum}(k) = \sum_{i=0}^{k} B[i]$.

If one chooses $\mathcal{D}$ as the Bernoulli distribution with probability of one being p, then Sum is intuitively similar to the Rank problem in an array of size $n = N$, containing $m = pn$ ones. Note, however, that in the Sum problem, the number of ones is a random variable. Thus, these two problems are identical only in the event $\mathcal{E} = \{\sum_{i=0}^{n-1} B[i] = pn\}$. Observe that $\Pr[\mathcal{E}] = \Omega(\frac{1}{\sqrt{pn}})$.

If one chooses $\mathcal{D}$ as the geometric distribution with success probability p, then SUM is intuitively similar to the SELECT problem in an array of size $n = N/p$, containing $m = N$ ones. Indeed, we can treat $B[i]$ as the distance between the ith and $(i+1)$th one in A. This works in the event $\mathcal{E} = \{\sum_{i=0}^{n-1} B[i] \leq N/p\}$. Observe that $\Pr[\mathcal{E}] = \frac{1}{2} \pm o(1)$: the skenewss of the geometric distribution is constant for p bounded away from one, so the Berry-Esseen theorem guarantees that the c.d.f. of $\sum_i B[i]$ is within ℓ_∞ distance $O(\frac{1}{\sqrt{n}})$ from the normal c.d.f.

The crucial ingredient of our lower bound was Lemma 3. We now state the following replacement for it:

Lemma 6. *Assume a data structure answers* SUM *queries on an array $B[0\,..\,N-1]$ of i.i.d. elements from $\mathcal{D}$, using P published bits, in the worst case, and $n \cdot \mathbf{H}(\mathcal{D})$ memory bits, on average. Assume (1) $\mathcal{D}$ is a Bernoulli with parameter p, and $P = o(pN)$; or (2) $\mathcal{D}$ is a geometric distribution with parameter p, and $P = o(N)$.*

Let $\gamma, \varepsilon > 0$ be appropriate constants. Break the queries into $k = \gamma \cdot P$ blocks. If:

$$\Pr_{A,q\in[n]}\Big[\text{Probes}(q) \cap \text{Probes}(Q_0) \neq \emptyset\Big] \;\leq\; \varepsilon,$$

there there exists an encoding for the input A of average size $n \cdot \mathbf{H}(\mathcal{D}) - \Omega(P)$.

Proof. Our proof of Lemma 3 only depended on A being a vector of uniform bits in one place: Fact 5. Fortunately, we have the following easy replacements:

Fact 7. *Let Z_n be the sum of n independent variables, sampled from a Bernoulli distribution with parameter p. Then, $\frac{1}{\sqrt{n\cdot p(1-p)}} Z_n - pn$ converges uniformly to the standard normal distribution $\mathcal{N}(0,1)$: the pointwise (ℓ_∞) distance between the two cumulative distribution functions is $O(\frac{1}{\sqrt{pn}})$.*

As before, this is an instantiation of the Berry-Esseen theorem, since the standard deviation of a Bernoulli trial is $\Theta(\sqrt{p})$ and the third absolute moment is $\Theta(p)$.

Fact 8. *Let Z_n is the sum of n independent variables, sampled from a geometric distribution with parameter p. Then, $\frac{p}{\sqrt{n(1-p)}} Z_n - \frac{n}{p}$ converges uniformly to the standard normal distribution $\mathcal{N}(0,1)$: the pointwise (ℓ_∞) distance between the two cumulative distribution functions is $O(\frac{1}{\sqrt{n}})$.*

This is another application of the Berry-Esseen theorem, using that the standard deviation of the geometric distribution is $\frac{\sqrt{1-p}}{p}$, and the third absolute moment is $O(1/p^3)$.

Our proof required $o(1)$ distance from the normal distribution. We apply these facts with $n = \omega(N/k) = \omega(N/P)$, so we need the technical conditions: (1) for the Bernoulli distribution, $N/P = \omega(1/p)$, or $P = o(pN)$; (2) for the geometric distribution, $N/P = \omega(1)$, or $P = o(N)$. □

The final trick we need is a way to deal with the events $\mathcal{E}$. As noted above, our $\sum$ problem is only identical to the RANK/SELECT problem under $\mathcal{E}$. The data structure assumed by

Lemma 6 only exists under the event $\mathcal{E}$, since we start with data structures for RANK, respectively SELECT.

To encode the input array B, we have two cases. If $\mathcal{E}$ is not satisfied, we simply spell out B with an efficient code, taking $n \cdot \mathbf{H}(\mathcal{D}) + O(1)$ bits. If $\mathcal{E}$ is satisfied, we apply the lemma, and obtain an encoding of size $n \cdot \mathbf{H}(\mathcal{D}) - \Omega(P)$. The average size will be $n \cdot \mathbf{H}(\mathcal{D}) + O(1) - \Omega(P) \cdot \Pr[\mathcal{E}]$.

For RANK, $\Pr[\mathcal{E}] = \Omega(\frac{1}{\sqrt{pN}})$, so we need to start with $P = \max\left\{r, \omega(\sqrt{pN})\right\}$ to always get a contradiction. The lemma applies as long as $P = o(pN)$. Thus, we get the lower bound $\max\left\{r, \omega(\sqrt{pN})\right\} \cdot w^{O(t)} \geq pN$. But this simplifies to $r \cdot w^{O(t)} \geq pN$, since for $r < (pN)^{1-\varepsilon}$, the lower bound on t only varies by constants. Since $\mathcal{D}$ is the Bernoulli distribution, $\mathbf{H}(\mathcal{D}) = H_2(p)$, where $h(\cdot)$ is the binary entropy function: $H_2(p) = p \cdot \log_2 \frac{1}{p} + (1-p)\log_2 \frac{1}{1-p}$. We have $\log_2 \binom{n}{pn} = n \cdot H_2(p) - O(\lg n)$, and $m = pn$. Thus, we obtain that any data structure using $\lg \binom{n}{m} + r$ bits and cell-probe complexity t needs to satisfy $r \cdot w^{O(t)} = \Omega(m)$.

For SELECT, $\Pr[\mathcal{E}]$ is constant, so we only need to start with P a big enough constant. The lemma applies as long as $P = o(N) = o(m)$. Since $\mathcal{D}$ is the geometric distribution with rate p, it has entropy $\mathbf{H}(\mathcal{D}) = \frac{1}{p} \cdot H_2(p)$. Thus, $N \cdot \mathbf{H}(\mathcal{D}) = \frac{N}{p} \cdot H_2(p) = n \cdot H_2(p) = log_2 \binom{n}{m} + O(\lg n)$. We again obtain that any data structure using $\lg \binom{n}{m} + r$ bits needs to satisfy $r \cdot w^{O(t)} = \Omega(m)$.

References

[CM96] David R. Clark and J. Ian Munro. Efficient suffix trees on secondary storage. In *Proc. 7th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 383–391, 1996.

[GGG+07] Alexander Golynski, Roberto Grossi, Ankur Gupta, Rajeev Raman, and S. Srinivasa Rao. On the size of succinct indices. In *Proc. 15th European Symposium on Algorithms (ESA)*, pages 371–382, 2007.

[GM03] Anna Gál and Peter Bro Miltersen. The cell probe complexity of succinct data structures. In *Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 332–344, 2003.

[Gol07] Alexander Golynski. Optimal lower bounds for rank and select indexes. *Theoretical Computer Science*, 387(3):348–359, 2007. See also ICALP'06.

[Gol09] Alexander Golynski. Cell probe lower bounds for succinct data structures. In *Proc. 20th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 625–634, 2009.

[GRR08] Alexander Golynski, Rajeev Raman, and S. Srinivasa Rao. On the redundancy of succinct data structures. In *Proc. 11th Scandinavian Workshop on Algorithm Theory (SWAT)*, 2008.

[Jac89] Guy Jacobson. Space-efficient static trees and graphs. In *Proc. 30th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 549–554, 1989.

[Mil99] Peter Bro Miltersen. Cell probe complexity - a survey. In *Proc. 19th Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 1999. Advances in Data Structures Workshop.

[Mil05] Peter Bro Miltersen. Lower bounds on the size of selection and rank indexes. In *Proc. 16th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 11–12, 2005.

[MRR01] J. Ian Munro, Venkatesh Raman, and S. Srinivasa Rao. Space efficient suffix trees. *Journal of Algorithms*, 39(2):205–222, 2001. See also FSTTCS'98.

[Mun96] J. Ian Munro. Tables. In *Proc. 16th Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 37–40, 1996.

[Pag01] Rasmus Pagh. Low redundancy in static dictionaries with constant query time. *SIAM Journal on Computing*, 31(2):353–363, 2001. See also ICALP'99.

[Păt08] Mihai Pătraşcu. Succincter. In *Proc. 49th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 305–313, 2008.

[PT06] Mihai Pătraşcu and Mikkel Thorup. Time-space trade-offs for predecessor search. In *Proc. 38th ACM Symposium on Theory of Computing (STOC)*, pages 232–240, 2006.

[PT07] Mihai Pătraşcu and Mikkel Thorup. Randomization does not help searching predecessors. In *Proc. 18th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 555–564, 2007.

[RRR02] Rajeev Raman, Venkatesh Raman, and S. Srinivasa Rao. Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In *Proc. 13th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 233–242, 2002.

[Vio09a] Emanuele Viola. Bit-probe lower bounds for succinct data structures. In *Proc. 41st ACM Symposium on Theory of Computing (STOC)*, pages 475–482, 2009.

[Vio09b] Emanuele Viola. Cell-probe lower bounds for prefix sums. arXiv:0906.1370v1, 2009.